AMENDMENTS TO THE CLAIMS

1. (Currently amended) A method of generating an intermediate representation during translation of subject code into target code, comprising the computer-implemented steps of:

decoding a plurality of instructions in the subject code;

generating an intermediate representation of the decoded instructions including providing a plurality of ~~IR-~~nodes in the intermediate representation as abstract representations of the expressions, calculations, and operations performed by the instructions of the subject code selected from a plurality of possible types of ~~IR-~~nodes including at least base nodes and complex nodes, wherein the base nodes represent the most basic semantics of the subject code such that the semantics of base nodes cannot be decomposed into other nodes representing more simple semantics, and wherein the complex nodes provide a more compact representation of the semantics of complex instructions in the program code than that of base node representations; and

determining at least one type of ~~IR-~~node out of the plurality of possible types of ~~IR-~~nodes to generate in the intermediate representation for each respective instruction in the decoded subject code.

2. (Canceled)

3. (Canceled)

4. (Currently amended) The method of claim 1, wherein the base nodes are generic across a plurality of possible subject architectures.

5. (Canceled)

6. (Currently amended) The method of claim 1, wherein the complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

7. (Previously presented) The method of claim 1, wherein a complex node may be decomposed into a plurality of base nodes to represent the same semantics of an instruction in the decoded program code.

8. (Currently amended) The method of claim 1, wherein the program code is designed to be executed by a subject architecture, the method further comprising the step of generating the complex nodes only for those features correspondingly configurable on the subject architecture.

9. (Currently amended) The method of claim 1, wherein the plurality of possible types of ~~IR-~~nodes further include polymorphic nodes.

10. (Currently amended) The method of claim 9, wherein the subject code is designed for execution on a subject architecture and is dynamically translated into the target code for execution on a target architecture, said method further comprising:

generating the intermediate representation to include the polymorphic nodes, wherein the polymorphic nodes each contain a function pointer to a function of the target architecture specific to a particular instruction in the subject code.

11. (Currently amended) The method of claim 10, said method further comprising generating the polymorphic nodes when the features of the target architecture would cause the semantics of a particular subject instruction to be lost if realized as base nodes.

12. (Original) The method of claim 10, wherein each polymorphic node is specific to a combination of a particular instruction in the subject code and a function of the target architecture.

13. (Currently amended) The method of claim 10, wherein:

said determining step further comprises identifying an instruction in subject code which corresponds an instruction on a list of polymorphic instructions to be realized as the polymorphic nodes; and

said generating step generates the polymorphic nodes only for those subject instructions corresponding to those on the list of polymorphic instructions, when a subject instruction corresponds to an instruction on the list of polymorphic instructions, when a subject instruction corresponds to an instruction on the list of polymorphic instructions.

14. (Currently amended) The method of claim 1, wherein the plurality of possible types of ~~IR-~~nodes further include architecture specific nodes.

15. (Previously presented) The method of claim 14, wherein the subject code is designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said method further comprising:

generating the intermediate representation to include architecture specific nodes which are specific to a particular combination of a subject architecture and a target architecture.

16. (Previously presented) The method of claim 15, wherein the generating step further comprises:

initially representing all of the instructions in the subject code as subject architecture specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject code;

determining whether an instruction in the subject code is one in which to provide a target architecture specialized conversion function;

converting subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generating base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation function.

17. (Original) The method of claim 16, further comprising generating corresponding target code from the target architecture specific nodes which is specialized for the target architecture.

18. (Original) The method of claim 15, further comprising generating corresponding target code from the base nodes which is not specialized for the target architecture.

19. (Canceled)

20. (Currently amended) A computer readable storage medium having translator software resident thereon in the form of computer readable code executable by a computer to perform the following steps during translation of subject program code to target program code:

decoding a plurality of instructions in the subject code;

generating an intermediate representation of the decoded instructions including providing a plurality of ~~IR~~-nodes in the intermediate representation as abstract representations of the expressions, calculations, and operations performed by the instructions of the subject code selected from a plurality of possible types of ~~IR~~-nodes including at least base nodes and complex nodes, wherein the base nodes represent the most basic semantics of the subject code such that the semantics of base nodes cannot be decomposed into other nodes representing more simple semantics, and wherein the complex nodes provide a more compact representation of the semantics of complex instructions in the program code than that of base node representations; and

determining at least one type of ~~IR~~-node out of the plurality of possible types of ~~IR~~-nodes to generate in the intermediate representation for each respective instruction in the decoded subject code.

21. (Canceled)

22. (Canceled)

23. (Currently amended) The computer readable storage medium of claim 20, wherein the base nodes are generic across a plurality of possible subject architectures.

24. (Canceled)

25. (Currently amended) The computer readable storage medium of claim 20, wherein the complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

26. (Previously presented) The computer readable storage medium of claim 20, wherein a complex node may be decomposed into a plurality of base nodes to represent the same semantics of an instruction in the decoded program code.

27. (Currently amended) The computer readable storage medium of claim 20, wherein the subject program code is designed to be executed by a subject architecture, the method further comprising the step of generating the complex nodes only for those features correspondingly configurable on the subject architecture.

28. (Currently amended) The computer readable storage medium of claim 20, wherein the plurality of possible types of ~~IR~~ nodes further include polymorphic nodes.

29. (Currently amended) The computer readable storage medium of claim 28, wherein the subject program code is designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said translator software further containing computer readable code executable by a computer to perform the following steps:

generating the intermediate representation to include the polymorphic nodes,

wherein the polymorphic nodes contain a function pointer to a function of the target architecture specific to a particular instruction in the subject code.

30. (Currently amended) The computer readable storage medium of claim 29, said translator software further containing computer readable code executable by a computer to generate the polymorphic nodes when the features of the target architecture would cause the semantics of a particular subject instruction to be lost if realized as base nodes.

31. (Original) The computer readable storage medium of claim 29, wherein each polymorphic node is specific to a combination of a particular instruction in the subject code and a function of the target architecture.

32. (Currently amended) The computer readable storage medium of claim 29, wherein said computer readable code executable by a computer for determining the type of ~~IR-~~nodes further:

identifies an instruction in subject code which corresponds an instruction on a list of polymorphic instructions to be realized as the polymorphic nodes; and

generates the polymorphic nodes only for those subject instructions corresponding to those on the list of polymorphic instructions, when a subject instruction corresponds to an instruction on the list of polymorphic instructions.

33. (Currently amended) The computer readable storage medium of claim 20, wherein the plurality of possible types of ~~IR-~~nodes further include architecture specific nodes.

34. (Original) The computer readable storage medium of claim 33, wherein the subject program code is designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said translator software further containing computer readable code executable by a computer to perform the following steps:

generating the intermediate representation to include architecture specific nodes which are specific to a particular combination of a subject architecture and a target architecture.

35. (Original) The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to perform the following steps:

initially representing all of the instructions in the subject code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject code;

determining whether an instruction in the subject code is one in which to provide a target architecture specialized conversion function;

converting subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generating base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation 14 function.

36. (Original) The computer readable storage medium of claim 35, said translator software further containing computer readable code executable by a computer to generate corresponding target code from the target architecture specific nodes which is specialized for the target architecture.

37. (Original) The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to generate corresponding target code from the base nodes which is not specialized for the target architecture.

38. (Currently amended) A translator apparatus for use in a target computing environment having a processor and a memory coupled to the processor for translating subject program code appropriate in a subject computing environment to produce target program code appropriate to the target computing environment, the translator apparatus comprising:

a decoding mechanism configured to decode instructions in the subject program code;

an intermediate representation generating mechanism configured to generate an intermediate representation of the decoded instructions including providing a plurality of ~~IR~~ nodes in the intermediate representation as abstract representations of the expressions, calculations, and operations performed by the instructions of the subject program code selected from a plurality of possible types of ~~IR~~-nodes including at least base nodes and complex nodes, wherein the base nodes represent the most basic semantics of the subject program code such that the semantics of the base nodes cannot be decomposed into other nodes representing more simple semantics, and wherein the complex nodes provide a more compact representation of the semantics of complex instructions in the subject program code than that of base node representations; and

an intermediate representation type determining mechanism configured to determine which type of ~~IR~~-nodes to generate in the intermediate representation for each respective instruction in the decoded subject program code.

39. (Canceled)

40. (Canceled)

41. (Currently amended) The translator apparatus of claim 38, wherein the base nodes are generic across a plurality of possible subject architectures.

42. (Canceled)

43. (Currently amended) The translator apparatus of claim 38, wherein the complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

44. (Previously presented) The translator apparatus of claim 38, wherein a complex node may be decomposed into a plurality of base nodes to represent the same semantics of an instruction in the decoded program code.

45. (Currently amended) The translator apparatus of claim 38, wherein the program code is designed to be executed by a subject architecture, the intermediate representation generating mechanism further comprising a complex node generating mechanism for generating the complex nodes only for those features correspondingly configurable on the subject architecture.

46. (Currently amended) The translator apparatus of claim 38, wherein the plurality of possible types of ~~IR~~ nodes further include polymorphic nodes.

47. (Currently amended) The translator apparatus of claim 46, wherein the subject program code designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, the intermediate representation generating mechanism further comprising:

a polymorphic node generating mechanism for generating the intermediate representation to include the polymorphic nodes,

wherein the polymorphic nodes contain a function pointer to a function of the target architecture specific to a particular instruction in the subject code.

48. (Currently amended) The translator apparatus of claim 47, said polymorphic node generating mechanism generating the polymorphic nodes when the features of the target architecture would cause the semantics of a particular subject instruction to be lost if realized as base nodes.

49. (Original) The translator apparatus of claim 47, wherein each polymorphic node is specific to a combination of a particular instruction in the subject code and a function of the target architecture.

50. (Currently amended) The translator apparatus of claim 47, wherein said intermediate representation type determining mechanism further comprises a polymorphic identification mechanism for identifying an instruction in subject code which corresponds an instruction on a list of polymorphic instructions to be realized as the polymorphic nodes; and

when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating mechanism generates the polymorphic nodes only for those subject instructions corresponding to those on the list of polymorphic instructions.

51. (Currently amended) The translator apparatus of claim 38, wherein the plurality of possible types of ~~IR~~ nodes further include architecture specific nodes.

52. (Previously presented) The translator apparatus of claim 51, wherein the subject program code is designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said intermediate representation generating mechanism further comprising:

an architecture specific node generating mechanism for generating the intermediate representation to include architecture specific nodes which are specific to a particular combination of a subject architecture and a target architecture.

53. (Previously presented) The translator apparatus of claim 52, the intermediate representation generating mechanism being configured to:

initially represent all of the instructions in the subject program code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject program code;

determine whether an instruction in the subject program code is one in which to provide a target architecture specialized conversion function;

convert subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generate base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation function.

54. (Previously presented) The translator apparatus of claim 52, further comprising a specialized target code generating mechanism for generating corresponding target code from the target architecture specific nodes which is specialized for the target architecture.

55. (Original) The translator apparatus of claim 52, further comprising a non specialized target code generating mechanism for generating corresponding target code from the base nodes which is not specialized for the target architecture.

56. (Original) The translator apparatus of claim 47, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

57. (Original) The translator apparatus of claim 47, wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.

58. (Original) The translator apparatus of claim 50, wherein when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating mechanism generates either polymorphic nodes or base nodes for those subject instructions corresponding to those on the list of polymorphic instructions.

59-79 (Canceled)

80. (Previously presented) The method of claim 10, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

81. (Previously presented) The method of claim 10, wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.

82. (Previously presented) The method of claim 13, wherein when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating step generates either polymorphic nodes or base nodes for those subject instructions corresponding to those on the list of polymorphic instructions.

83. (Previously presented) The method of claim 1, wherein the translation is a dynamic binary translation from the subject code as binary machine code of a subject instruction set architecture into the target code as binary machine code of a target instruction set architecture.

84. (Previously presented) The computer-readable storage medium of claim 29, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

85. (Previously presented) The computer-readable storage medium of claim 29 wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.